Partial Tokenizer

Michael Russo
Net ID: mr880

Read Me:
The program utilizes functions to pursue the logic we desire.
Functions include:

1. int escapeChars(char* tk) This function traverses a string in search of escape functions one would enter into a program like /a, /b, /v etc. and identifies them to the user as an error.

2. void TKDestroy(TokenizerT * tk) This function frees up the memory from TKCreate and is run at the close of our main.

3. TokenizerT *TKCreate(char * ts) This function sets aside the memory and copies in data from the user.

4. int malCheck(char* tk) In the malCheck function, we check our input for tokens that represent unfinished or poorly formed "would-be" tokens and identifies them to the user.

5. int malcheckAlt(char *tk) An alternate mal check function used for identifying the special case of a mal token in which the values are would-be hexadecimal tokens.

6. int zeroCheck(char* tk) This function tests for cases in which a zero is present without the hexadecimal conclusion, octal conclusion or float conclusion.

7. int floatToken(char *tk) The float function checks for values that come out as a float

8. int decimal(char* tk) The decimal checks for decimal values

9. int octalCheck(char *tk) The octal checks for octal values

10. int hexCheck(char *tk) The Hex checks for hexadecimal values

11. char* TKGetNextToken(TokenizerT * tk) Our TKgetNextToken function performs a very function in which our tokens are ordered and outputted in a specific order that allows for no errors.

Conclusion: The program runs with very few to zero errors. While heavy, it performs its tasks well and gets the job done.